Re-Exam in Algorithms and Data Structures 1 (1DL210)

Department of Information Technology
Uppsala University
2014–10–20
Lecturers: Mohamed Faouzi Atig, Tuan Phong Ngo, Jari Stenman and Yunyun Zhu
Location: Fyrislundsgatan 80, Exam Hall 1
Time: 08:00 - 13:00
No books or calculator allowed

Directions:

1. Do not write on the back of the paper
2. Write your anonymous code on each sheet of paper
3. **Important** Unless explicitly stated otherwise, justify you answer carefully! Answers without justification do not give any credits.

Good Luck!

Problem 1 (10p)

a) Order these functions in order of increasing asymptotic growth rate [1]. If two of them have the same asymptotic growth rate, state that fact. No justification is needed.

$$\left(\frac{3}{2}\right)^n \qquad 4^n \qquad \left(\frac{2}{5}\right)^n \qquad 3^{3-n} \qquad n^3 \qquad log(n^3) \qquad n^3+n\,log(n) \qquad n$$

b) State whether the following statements are true or false. No explanation is needed.

(ii) The worst case running time of QuickSort is $O(n!)$

(iii) $3^n + n^3 = \Omega(4^n)$.

(iv) $3^n + n^3 = \mathcal{O}(2^n)$.

---

[1] Here, *log* denotes the binary logarithm.

Solution 1 a) The order from left to right is the following:

$$\left(\frac{2}{5}\right)^n \qquad 3^{3-n} \qquad log(n^3) \qquad n \qquad n^3 \qquad n^3{+}n\,log(n) \qquad \left(\frac{3}{2}\right)^n \qquad 4^n$$

Furthermore, n^3 and $n^3 + n\,log(n)$ have the same order of growth.

b) State whether the following statements are true or false. No explanation is needed.

(ii) true

(iii) false.

(iv) false.

Problem 2 (12p) Consider Insertion-Sort, Merge-Sort and Heap-Sort. For each algorithm, what will be the worst case asymptotic upper-bound on the running time if you know additionally that

a) the input is already sorted?

b) the input is reversely sorted?

c) the input is a list containing only n copies of the same number?

For each case, state your answer and justify it.

Solution 2 (12p)

	Insertion-Sort	Merge-Sort	Heap-Sort
sorted array	$O(n)$	$O(nlog(n))$	$O(nlog(n))$
reverse sorted array	$O(n^2)$	$O(nlog(n))$	$O(nlog(n))$
the same number	$O(n)$	$O(nlog(n))$	$O(nlog(n))$

These answers follows directly from the best and worst cases of these sorting algorithms (see the lecture notes). In your answers you should give more details concerning the justifications.

Problem 3 (8pt) Let A be an unsorted array of size n. Give an algorithm that finds a pair $i, j \in A$ such that it minimizes the **absolute** value of $(A[i] - A[j])$ for $i \neq j$. Example 1: if the array is given by $[1, 4, 2, 3]$ then a possible values of i and j are 1 and 3, respectively. Example 2: if the array is given by $[0, 6, 3, 8]$ then a possible values of i and j are 2 and 4, respectively. Your algorithm should run in $O(nlgn)$ in the worst case.

Solution 3 The algorithm proceeds as follows:

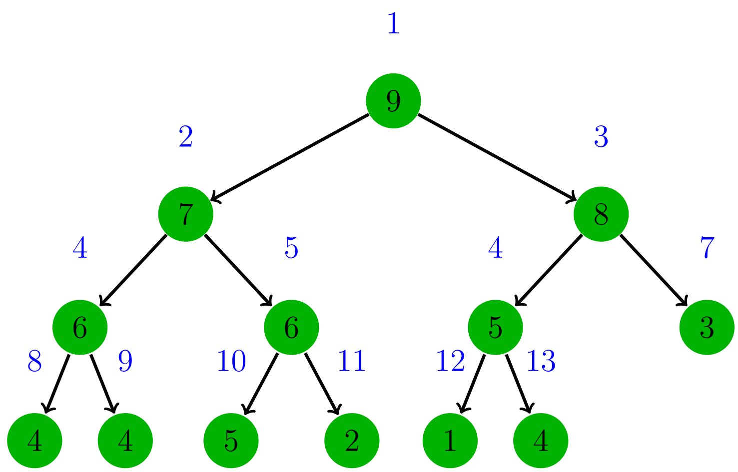
- Make a copy of the array A (let us call it B). This can be done in $O(n)$.
- Sort the array B. This can be done in $O(nlog(n))$ using merge sort.
- Compute an array C of size $n-1$ such that each position $k \in \{1, ...n\}$ contains $B[k+1] - B[k]$. This can be done in $O(n)$.
- Design an algorithm that returns the position $k_{min} \in \{1, ..., n-1\}$ in the array C that contains the smallest value in the array C. This can be also done in $O(n)$.
- Find and return the index i and j in the array A such that $A[i] = B[k_{min}]$ and $A[i] = B[k_{min}+1]$, respectively. This can be done in $O(n)$.

Finally, summing up all the time complexities, we obtain an algorithm that takes $O(nlog(n))$.

Problem 4 (10p)

Give the max-heap that results when the keys
`6 4 1 5 8 9 3 4 6 7 2 4 5`
are inserted into an initially empty max-heap.

Solution 4 Below is the resulting max-heap:

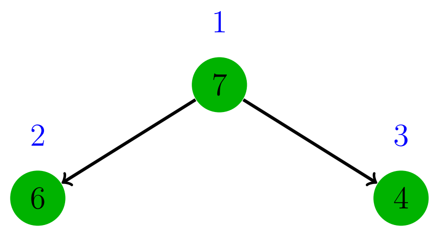


Problem 5 (10p) Assume that we have a max-heap H and an integer x such that $x < \text{HEAP-MAXIMUM}(H)$. Assume also that we construct the heaps $H_1, \ldots, H_4$ in the following way:

- Let H_1 be the resulting heap after executing $\text{MAX-HEAP-INSERT}(H, x)$.
- Let H_2 be the result of executing $\text{HEAP-EXTRACT-MAX}(H_1)$.
- Let H_3 be the resulting heap after executing $\text{HEAP-EXTRACT-MAX}(H)$.
- Let H_4 be the result of executing $\text{MAX-HEAP-INSERT}(H_3, x)$.

Is it always the case that $H_2 = H_4$? In other words, does it matter in which order we do MAX-HEAP-INSERT and HEAP-EXTRACT-MAX, as long as we do not remove the value we just inserted? If your answer is yes, justify in no more than 5 lines. If your answer is no, give a counterexample by providing concrete values for H and x and drawing H_1, H_2, H_3 and H_4.

Solution 5 The answer is **no**. A possible counter example can be constructed with $x = 5$ and the following max-heap:



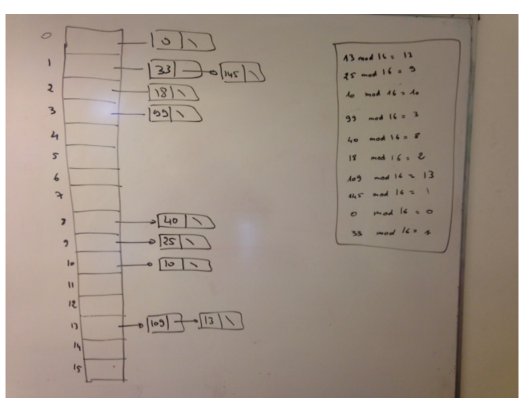
Problem 6 (10p)

Consider inserting the following keys into a hash table of length $m = 16$, in the order they are listed (first 13, then 25, and so on):

$$13 \;\; 25 \;\; 10 \;\; 99 \;\; 40 \;\; 18 \;\; 109 \;\; 145 \;\; 0 \;\; 33$$

The auxiliary hash function is given by $(k \mod m)$. Draw the resulting hash table if we use chaining to resolve collisions.

Solution 6 You need first to specify if the first index of the hash table is 0 or 1. Below I have considered that the first index of the hash table is 0.



Problem 7 (10p) Consider two hash functions defined as:

$$\begin{aligned} h_1(k) &= k \mod n \\ h_2(k) &= 2 \cdot k \mod n \end{aligned}$$

Explain in whether one of h_1, h_2 will perform better than the other in practice if

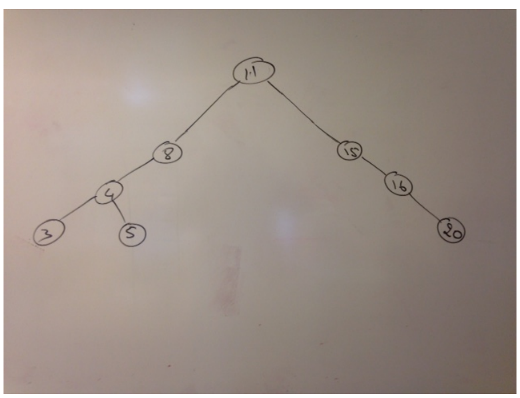
a) n is even

b) n is odd

Solution 7 When n is even, h_1 performs better than h_2 since h_2 will map all the keys to the even slots and so the half of the hash table will never be used. When n is odd, h_1 and h_2 would have the same performance since all the slots will be mapped.

Problem 8 (10p) Suppose that we start from an empty binary search tree and insert the following elements: 10,11,9,8,15,16,4,3,20,5. Then, we delete the elements: 9 and 10. Show the tree you obtain after each insertion/deletion.

Solution 8 Below I give only the resulting BST but you should show all the all the trees after each operation:



Problem 9 (10p)

Suppose that we have numbers between 1 and 100 in a binary search tree and want to search for the number 45. Which of the following sequences could **not** be the sequence of nodes examined?

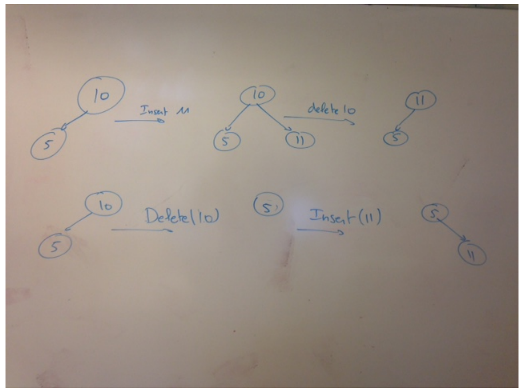
- 34, 35, 36, 50, 33, 39, 77.
- 1, 2, 3, 5, 6, 7, 18, 29.
- 59, 48, 36, 37, 4, 3, 2, 1.
- 2, 9, 37, 19, 36, 82, 31, 78.

Solution 9 Below some short answers:

- 34, 35, 36, 50, 33, 39, 77. It is not good searching sequence since we have chosen to move the right subtree from 34 but later on we have encountered 33 which is smaller than 34.
- 1, 2, 3, 5, 6, 7, 18, 29. It is a good searching sequence.
- 59, 48, 36, 37, 4, 3, 2, 1. It is not good searching sequence since we have chosen to move right from 36 but later on we have encountered 4 which is smaller than 36.
- 2, 9, 37, 19, 36, 82, 31, 78. It is not good searching sequence since we have chosen to move left from 37 but later on we have encountered 82 which is larger than 34.

Problem 10 (10pt)
Is the operations of insertion and deletion in a binary search tree commutative in the sense that deleting x and then inserting y from a binary search tree results in the same tree as inserting y and then deleting x? Argue why it is so or give a counter-example.

Solution 10 The answer is no. Here is a counter-example.


10
5
Insert 11
10
5
11
delete 10
11
5
10
5
Delete(10)
5
Insert(11)
5
11